Process Assessment Issues in a Bachelor Capstone Project

Vincent Ribaud
Univ. Bretagne Occidentale,
UMR 6285, Lab-STICC
CS 93837, F-29200 Brest, France
ribaud@univ-brest.fr

Alexandre Bescond, Matthieu Gourvenec, Joël Gueguen, Victorien Lamour, Alexandre Levieux, Thomas Parvillers
Univ. Bretagne Occidentale, FR,
Bachelor of Computer Science
{FirstName.LastName}@
etudiant.univ-brest.fr

Rory V. O'Connor
School of Computing, Dublin City University
Glasnevin, Dublin 9, Ireland
roconnor@computing.dcu.ie

Abstract

Based on a small subset of ISO/IEC 15504:2006, a Process Assessment was performed in the capstone project of a Bachelor in Computer Science. Parallel to this assessment, students performed a continuous self-assessment using an ability model based on 15504 Base Practices and Work Products. This paper highlights how students' self-assessment and teacher's assessment are correlated. The capstone project itself implements major constructivism principles. This paper presents also the students' point of view through different questionnaires and students' participation to the paper.

1. Introduction

The ACM Computing Curricula [ACM05] establishes the following requirement for a Bachelor curriculum: *"Demonstration that each student has integrated the various elements of the undergraduate experience by undertaking, completing, and presenting a capstone project."* The capstone project is intended to provide students with a learning by doing approach about software development, from requirements to qualification testing. Indeed, the project progress is sustained by software processes. It helps students to be conscious about and improve what they are doing when processes are replaced in a whole picture and when a continuous assessment provide them with objective feedback. Hence, a main capstone teacher's activity is to assist students with appreciation and guidance, a task that relies on the assessment of students' practices and students' products. Students were encouraged to perform a self-assessment in parallel of the teacher's assessment. Consequently, we implemented an experimental protocol to observe how students' self-assessment and teacher's assessment are correlated.

Proceedings of the International Workshop on Software Process Education, Training and Professionalism, Gothenburg, Sweden

20015-06-15 published at http://ceur-ws.org

Our implementation of a capstone project results from a twenty years experience about project and problem-based learning for software development. From the designer's side - the teacher, most constructivism principles are taken in account and implemented. However, what's up from the constructors' side – the students – The question was raised to the class using several questionnaires and several students accepted to anonymize answers and to analyze results. Hence they are co-authors of this paper whose structure is: section II presents process assessment, section III statistics and pedagogical practices, section IV the practicum, students and teacher roles. Questionnaires results are intertwined in the sections and commented by students and teacher.

2. Process assessment

The main goal of the capstone project is to learn by doing a simplified cycle of software development through a somewhat realistic project. Until this year, students worked in small teams (2-3 people). Thanks to doubling the hours allocated to the project this year and to avoid too much behaviorist division of labor between students, the capstone project was performed individually from A to Z.

2.1 Software processes

A side-effect goal of the capstone project is to be exposed to some kind of process assessment. We choose a small subset of the ISO/IEC 15504:2006 Process Reference Model, mainly the Software-related Processes of the ENG Process Group [15504-Part 5]: ENG.3 System architectural design, ENG.4 Software requirements analysis, ENG.5 Software design, ENG.6 Software construction, ENG.7 Software integration, ENG.8 Software testing. Process Purpose, Process Objectives and Base Practices have been kept without any modification; Input and Outputs Work Products have been reduced to the main products.
We recall some definitions from the ISO/IEC 15504 standard [15504]: "processes are grouped according to the type of activity they address: the processes included

in the same group contribute to a complementary area", "a process is a set of interrelated or interacting activities which transforms inputs into outputs", "a base practice is an activity that, when consistently performed, contributes to achieving a specific process purpose", and "a work product is an artifact associated with the execution of a process".

2.2 Ability model

From an individual human perspective, the 15504 Exemplar Process Model can be seen as a competencies model related to the knowledge, skills and attitudes involved in a software project. A competencies model defines and organizes the elements of a curriculum (or a professional baseline) and their relationships. During the education period, all the students use the same model but they can individually change afterwards.

A hierarchical model is easier to manage and use. We kept the hierarchical decomposition issued from the 15504: process groups –process – base practices and products. A competency model is decomposed into competency areas (mapping to process groups); each area roughly corresponding to one of the main division of the profession or of a curriculum. Each area organizes the competencies into families (mapping to processes). A family roughly corresponds to main activities of the area. Each family is made of a set of knowledge and abilities (mapping to base practices), eventually called competencies; each of these entities being represented by a designation and a detailed description.

2.3 Process Assessment

ISO 15504 [15504] defines a measurement framework for the assessment of process capability defined on a six point ordinal scale which represents increasing capability of the implemented process, from not achieving the process purpose through to meeting current and projected business goals. [15504-2]. Within this measurement framework, the measure of capability is based upon a set of process attributes (PA). Each attribute defines a particular aspect of process capability. The extent of process attribute achievement is characterized on a defined rating scale: N Not Achieved, P Partially Achieved, L Largely Achieved, F Fully Achieved. Capability Level 0 denotes an incomplete process, either not performed at all, or for which there is little or no evidence of systematic achievement of the process purpose [15504-3]. Capability Level 1 denotes a performed process that achieves its process purpose through the performance of necessary actions and the presence of appropriate input and output work products which, collectively, ensure that the process purpose is achieved [15504-3]. Higher levels denote higher process maturity: the process is managed (Level 2), established (Level 3), predictable (Level 4), optimizing (Level 5).

If students are able to perform a process, it denotes a successful learning of software processes, and teachers' assessments rate this capability. Because we believe that learning is sustained by continuous assessment, self-directed, done by teachers or a third-party, the research question aims to state how students' self-assessment and teacher's assessment are correlated and if self-assessment for performing BP and delivering WP is an alternative to external assessment about 15504 Level 1. Obviously, the assessment main goal is students' ability to perform the selected processes set.

3. The capstone project

This section overviews the project and assessment results, then presents each process with assessment details, teacher's analysis and students' comments.

3.1 Overview

3.1.1 Schedule

The curriculum is a 3-year Bachelor of Computer Science. The project happens the third year before students' internship. The project is performed during a period of 2 weeks. Before the dedicated weeks, 40 lecture hours are dispatched all the semester along and some homework is required. Ideally, students should be familiar with the Author-Reader cycle as the project starts and have performed the requirements and architectural design processes. Each deliverable can be reviewed as much as needed by the teacher that provides students with comments and suggestions.

3.1.2 System architecture

The system is made of 2 sub-systems:

- PocketAgenda (PA) for address books and agenda management and interface with a central directory;
- WhoIsWho (WIW) for managing the directory and a social network.

PocketAgenda is implemented with Java, JSF relying on a Oracle RDBMS. WhoIsWho is implemented in C or Java using a small RDBMS or files. Both sub-systems communicate with a protocol to establish using UDP.

The system is delivered in two batches. Batch 1 scope is: PocketAgenda – address book and directory interface; WhoIsWho - directory management. Batch 2 scope is: PocketAgenda – agenda and social network interface; WhoIsWho – social network management.

3.1.3 Rating scheme

Table 1 presents the rating scheme. Students' assessment was continuous and communicated to students regularly; hence they have been made aware of their progression each day and adjusted their effort.

Table 1: Rating scheme

Process	**Work product**	**Pt.**
	Batch 1	
ENG.4	Use cases –Social network	1
ENG.3	Interfaces specification	1
ENG.5	Detailed Design Document	2
ENG.6	4GL applications	3
ENG.6	Network application	3
	Batch 2	
ENG.4	Use cases	3
ENG.6	4GL applications	2
ENG.6	Java/SQL application	1
	Project	
ENG.7	Integration schema, configuration, version sheet	1
ENG.8	Test reports	1
Attitude	Assiduity, commitment, organization	2
Total		**20**

3.1.4 Statistics

Table 2 presents teacher's assessment. BP and WP rating are aggregated using an all-or-none principle: if all BP or WP in a process are rated at least Largely (or Fully), the process is rated Largely (or Fully)[1]. At the two-third of the project, students have been made aware of the Level 2 and its attributes. However, the teacher has not enough time to track the PA 2.1 Performance management and only the PA 2.2 Work product management was tracked for the most advanced students: those who were assessed by the teacher for all processes at L or F; it represents 7 students over 23.

Table 2: Teacher's assessment

	BP level 1		*WP level 1*		*WP2*
	L	*F*	*L*	*F*	*L*
ENG.4 Requirement	3	13	7	10	3
ENG.3/5 Design	7	3	8	8	2
ENG.4 Construction	4	6	10	6	4
ENG.7 Integration	7	0	5	2	1
ENG.8 Testing	6	10	12	3	1

As the project ends, students have to complete a summary including hour's breakdown and self-assessment of achievement for each process. Summary was mandatory and 22 students over 23 completed it. Table 3 presents students' self-assessment and the average hours spent for each process. Last column indicates the number of times where the teacher's assessment matches the student's self-assessment.

Table 3: Overview of self-assessment and match

	Hrs	*N*	*P*	*L*	*F*	*Match*
ENG.4 Requirements	20	0	0	8	14	18
ENG.3/5 Design	19	0	0	11	11	16
ENG.4 Construction	48	0	6	9	7	14
ENG.7 Integration	9	4	7	9	2	11
ENG.8 Testing	5	2	7	9	4	10

3.1.5 Information about students

The class comprises 24 students. One gave up in the middle of the project. Among 23 remaining, 3 students' projects failed, 4 projects were barely satisfactory, 9 good, 5 very good and 2 excellent. Questionnaires were

[1] BPs that are a kind of Develop test criteria or Develop test procedures, are out of scope and excluded from aggregates.

completed by 22 students. 6 students have participated to the writing of this paper and were classified as: 1 project failed, 1 was barely satisfactory, 1 good, 2 very good and 1 excellent.

A unique student works in parallel. 20 completed first and second year in our Bachelor. 17 were assiduous. 15 repeated at least a class before the Bachelor final year (in high school or at the university). 15 were able to perform the project outside the university labs. 10 claimed to have a good knowledge of SQL and Java before the project.

3.2 Project progress

Students were advised that they can freely participate to the following experiment: they will have to regularly update a competency model comprising the ENG process group, the 6 processes above and their Base Practices and main Work Products and self-assess on the N-P-L-F scale. The teacher will also assess the same BPs and WPs and volunteers students will correlate self-assessment and teacher's assessment and deliver anonymous results for the paper. All students did agree with the experiment but only 18 delivered the completed competency model to volunteers. The data distribution is presented in tables in each process subsection. The match with teacher's assessment is indicated as the last column of each table. Teacher analysis and comments made by students co-authoring the paper are reported at end of process subsection.

3.2.1 Requirements

According to students' estimates average, they spent 20 hours over 102 total hours to capture, write and manage requirements through use cases. A 4-hour lecture about use cases was delivered in January at the beginning of the semester, then the iterative process of writing and being reviewed by the teacher started. When the project full-time period had started, 6-7 students over 22 have completed the requirement process and produced the requirement specification WP. The remaining completed theses tasks during the project. Without surprise, the more backward students (for this task or the following one) failed.

Table 4 presents main Base Practices (ENG.4.BP1: Specify software requirements; ENG.4.BP3: Develop criteria for software testing; ENG.4.BP4: Ensure consistency) and main Work Products (17-11 Software requirements) for the ENG.4 Software requirements analysis process.

Table 4: ENG.4 assessment (self and teacher)

	N	*P*	*L*	*F*	*Match*
BP1. SW requirements	0	2	9	7	6
BP3. Test criteria	0	5	7	5	2
BP4. Consistency	1	3	7	7	8
17-8 Interface requirements	0	3	1	8	6
17-11 SW requirements	0	1	10	7	9

Thanks to the Author-Reader cycle, specification writing iterates several time during the semester and the final mark given to almost SW requirement document was Fully Achieved. However matching between students and teacher assessments is poor. A deeper look on data yields a possible explanation: "good" students, despite the excellent final mark, were aware of the cycle and the improvement suggested by the teacher at each iteration, hence they self-assess generally as Largely Achieved whereas the teacher rated a Fully Achieved; "normal" students took the final mark as the level they achieved and self-assessed as F whereas the teacher rated a L.

Clearly, students did not understand the ENG.4.BP3: Develop criteria for software testing and failed the self-assessment. The definition is "*Use the software requirements to define acceptance criteria for the software product tests. Software product tests should demonstrate compliance with the software requirements*. [15504-Part 5]" The teacher defined acceptance criteria and students were not aware of this topic, however they confused "develop criteria for SW testing" and "testing SW" and self-assessed at a much higher level that the teacher did.

Students' comment. It was the first time that we have to write use cases from a statement of work. Eliciting and writing requirements were difficult and the Author-Reader cycle helped to produce complete and usable use cases and to acquire a writing style. Because of the novelty of the task and to achieve a certain maturity degree, it is required to start the writing task early in the semester.

3.2.2 Architectural and detailed design

On average, students spent 19 hours over 102 total hours to perform architectural and detailed design. Design is split in data modeling, Web-based design and

oriented-object design. The PocketAgenda subsystem is structured around a database schema. Modeling is performed using SQL Developer Data Modeler, freely available through the Oracle Academy program. Data architectural design results in a Logical model, data detailed design (obvious in that case) is performed through automated forward engineering and results in a Relational model. A 2-hour lecture about Data Modeler was delivered in February after the use cases phase. then the iterative Author/Reader cycle started.

Jdeveloper is a Java IDE for the Oracle Application Development Framework (ADF). ADF is an end-to-end development framework, built on top of the Enterprise Java platform, and providing integrated solutions including data access, business services development, a controller layer, a JSF tag library implementation. 12 labs hour were devoted to learning the framework, insufficient for mastering the IDE but enough for a quick start.

UML modeling and object-oriented design are taught in dedicated lectures during the curriculum (30 hours each). However, nearly all students had no idea how to perform the design. Design was taught by example: students have developed a component of the batch 1 from a design document provided by the teacher. Then they had to develop another batch 1 components and retro-design their development. Finally they had to establish the design of remaining components.

Architectural design was also shown by example: a complete cycle was provided for one networked function: use case, interface specification, design for the client and server sides, client and server stubs program. Students reproduced the scheme.

Table 5: ENG.3 and 5 assessments (self and teacher)

	N	*P*	*L*	*F*	*Match*
BP1. Describe syst. arch.	0	4	10	5	6
BP3. Define interfaces	0	4	6	8	6
BP3. Detailed design	0	2	9	7	7
BP4. Consistency	1	2	9	6	8
04-01 Database design	0	1	6	11	13
04-04 High level design	0	4	6	8	8
04-05 Low level design	0	2	8	8	11

Table 5 presents main Base Practices (ENG.3.BP3: Define interfaces; ENG.5.BP3: Develop detailed design; ENG.5.BP5: Ensure consistency) and main Work Products (04-01 Database design; 04-04/05 High/low level SW design) for the ENG.3 et 5 System and software design process.

Again, matching is poor, except maybe for technical design. A similar concern to requirements arose with design: a few students were aware of the improvement cycle performed by the Author-Reader cycle and took the Work Product (Design Document) as an indication of their achievement. Another explanation is related to the fact that bachelor students are focused on technology, hence there are more able to self-assess on technical tasks (Database or Detailed Design).

Students' comment. Requirement specifications greatly helped to figure out the system behavior and facilitated the design phase and interface specification. However, students had never learnt architectural design and interfaces between sub-systems. Design time has to be immediately followed by coding time and could not spread along the semester as we did it for requirements. Students performed high level design for a batch and low level design for the other, and both have advantages depending on the student's personality: either creative or preferring to be guided.

3.2.3 Construction

On average, students spent 48 hours over 102 total hours to develop the software. Java, network programming and database / SQL programming are taught in dedicated lectures during the curriculum (60 hours each). Despite of this amount, 12 students self-judged as having a poor knowledge of SQL and Java, and 10 students were unable to develop the client-server application although a Java server skeleton has been provided. Time constraints also played their role: because the network component was perceived by difficult by some students, they did not commit to the work and invested others more cost-effective tasks. Students have almost no idea of test-driven development and a lack of a test strategy; hence unit were poorly tested. This point has to be addressed in the next edition.

Table 6 presents main Base Practices (ENG.6.BP1: Develop unit verification procedures; ENG.6.BP2: Develop SW units; ENG.6.BP3: Ensure consistency; ENG.6.BP4: Verify SW units) and main Work Products (11-05 Software unit; 14-04 Test log) for the ENG.6 Construction process.

Table 6: ENG.6 assessment (self and teacher)

	N	*P*	*L*	*F*	*Match*
BP1. Verification procedures	2	3	10	3	6
BP2. Develop units	0	5	8	5	8
BP3. Consistency	0	7	8	3	9
BP4. Verify units	0	7	8	3	6
11-05 Software unit	0	4	9	5	8

Unit testing is a little more familiar to students, and although they probably misunderstood the ENG.6.BP1: Develop unit verification procedures; the matching is not so worse that for the ENG.4.BP3: Develop criteria for software testing. The discrepancy between students and teacher assessments about ENG.6.BP2: Develop software units stems from the "goggle-paste" phenomena; only a few students writes his/her own code and has been assessed at the Largely or Full level by the teacher; most students adapt code from others without a real understanding of the programming activity and over-assess themselves.

Students' comment. This process raised a certain anxiety because students had doubt about their ability to develop a stand-alone server interoperating with a JDeveloper application. Students had never learnt a 4GL (fourth generation language) environment such as JDeveloper, hence they reported that the switch from a 3GL to a 4GL was difficult but once understood, they appreciated the power leverage of such environments. The majority of students whose successfully developed the client-server component reported that they could not achieve it without the help of the skeleton provided by the teacher. For some students, a poor Java literacy prevent them to struggle with the network part. Some students failed because they jumped to code before having any draft or idea to realize it.

3.2.4 Integration and tests

On average, students spent 15 hours over 102 total hours to integrate and perform qualification tests of the software. These topics are unaddressed in the curriculum and because they mostly occur at the end of the project, no time was available to complete the learning. In the best cases, students have respected their interfaces specification and few problems arose when they had to integrate the Java client program within the JDeveloper application. In other cases, they were unable to perform the integration and the assessment was partial and based on the Java client code. Test cases specification stemmed from use cases, hence no test plan was required. Test procedure was reduced to test each use case - success scenario and main extensions, to verify the conformity to use cases and the results achieved.

Table 7 presents main Base Practices (ENG.7.BP3: Integrate software item; ENG.7.BP5: Ensure consistency; ENG.8.BP1: Develop tests for integrated software product; ENG.8.BP2: Test integrated software product) and main Work Products (08-21 Software test plan; 11-01 Software product; 14-04 Test log) for the ENG.7 et 8 Software integration and software testing process.

Table 7: ENG.7 and 8 assessments (self and teacher)

	N	*P*	*L*	*F*	*Match*
BP3. Integrate SW items	2	4	10	1	9
BP5. Consistency	2	3	11	1	9
BP1. Develop tests	2	5	10	2	4
BP2. Test product	0	5	9	3	9
08-21 Software test plan	0	4	11	2	3
11-01 SW product	0	4	9	4	9
14-04 Test log	0	4	11	2	10

We observe the same poor correlation for the ENG.8.BP1: Develop tests for integrated software product and the WP 08-21 Software test plan, indicating that students are not aware of the test definition and planning activity, a common hole in a Bachelor curriculum although testing is an ability strongly required by employers.

Integration is also an uncovered topic and students are not aware of the subject: for the ENG.7.BP3: Integrate software item, 11 students (over 18) were assessed by the teacher as Not or Partially whereas they only 6 self-assessed N or P.

Students' comment. Some students were aware of the poor maturity of the integrated product, partly due to the lack of testing. Although the Junit framework has been taught during the first semester, some students did not see the point to use it while some others did not see how to use it for the project. Students that did not develop the server had no integration to perform.

4. Students and teacher roles

Constructivism can be summed up with two fundamental statements [Duf96]: (i) learning is defined as an active process for knowledge building rather than a knowledge acquisition process; (ii) teaching is essentially aimed at helping students in this process rather than transmitting knowledge.

Among practices belonging to the constructivist stream, Dwyer [Dwy94] and Tardif [Tar98] define a learning paradigm, in opposition with the main teaching paradigm. The learning paradigm provides a framework which allows the school to constitute a learners' community for the pupils as well as the teachers and the other staff members.

This section aims to relate the educational system with the new roles required in a constructivism approach. The questionnaire collects anonymously students' perception about roles. Teacher's role has to be rated on the scale used to rate practices and products: Not achieved, Partially Achieved, Largely Achieved, Fully Achieved. Students' self-opinion about their roles and about the practicum are expressed on a 5-point Likert scale from Strongly Agree to Strongly Disagree.

4.1 Teachers' role

Tardif [Tar98] defines teachers' roles as creators of pedagogical environments; interdependent, open-minded, critical professionals; development instigators; mediators between knowledge and students; coaches; collaborators for the students' success of a whole school. The first role was questioned in a special part of the questionnaire related to the educational system and is presented in section 4.3. Table 8 presents students' rating about the teacher's roles.

Table 8: Students' rating about teacher's roles

	?	N	P	L	F
a professional, open-minded and open to criticism	1			4	17
a development instigator			1	8	13
a mediator between knowledge and students			2	7	13
a coach	1		4	6	11

Students' comment. Students agree with the teacher's roles required. The majority of students want to be instigated but not directed to a solution. Some students stated that teachers fall into two categories: those that don't care of students and those that help too much and deprive them of autonomy because they want to control the learning results. They appreciated the balanced teacher's attitude and to be on his or her own but also to have a teacher in case of emergency. Students noticed that the teacher wanted that everyone speak, discuss and compare points of view and aimed an active participation. Some students complained that the teacher did not share his time equally between students and pointed out that a second teacher will be useful.

4.2 Students' role

Tardif [Tar98] defines students' roles as investigators; co-operators sometimes experts; clarifying actors; strategic users of available resources. The questionnaire set the following definitions: *investigator*: I discussed with other students my questions about the project and/or I defended my solutions; *co-operators sometimes experts*: I explained some project points to other students and/or I had myself explanations from others; *clarifying actors*: I asked the teacher or other students in order to insure my good project understanding and to verify the adequacy of my proposals; *strategic users of available resources*: I used the available resources and/or supplementary resources and I verified their relevance. Table 9 presents students' perception about their roles.

Table 9: Students' self-perception about their roles

	strg agr	agr	neu-tral	dsgr	strg dsgr
investigator	12	8	2		
co-operators - experts	10	11		1	
clarifying actors	14	7	1		
strategic users	7	8	6		

Students' comment. Some students underestimate themselves and some definitions (strategic users, for instance) were seen as out of the reach and they could not use it to qualify themselves. However students have learnt to debate, find and explain solutions. Students learnt a lot about to work with consistency and traceability, to respond to demands within the recommended time and to log his or her work in order to notice the project progress.

4.3 The practicum

Tardif [Tar98] defines the characteristics of a pedagogical environment (the practicum) consistent with the learning paradigm: constancy of learning and time variability; cognitive imbalance; authenticity of learning situations; transdisciplinarity; interactions between theory and practice; embedment of assessment within the learning situations. The last part of the questionnaire let students express their opinions about the practicum, which are presented in Table 10.

Table 10: Students' self-perception about the practicum

The Agenda project	strg agr	agr	neu-tral	dsgr	strg dsgr
I had the time to learn and do the project.	6	11	2	3	
I found the project complex.	5	12	2	3	
I committed to perform the project.	14	6	1	1	
I found the project realistic.	11	8	1	1	1
I understand relationships between specifications, design, building and tests.	15	6	1		
I had to deepen my knowledge and skills to perform the project.	10	12			
My work for the project helped me to understand lectures.	5	6	8	1	2
I used a lot the reviewing facilities.	7	7	1	5	2
I made progress thanks to the reviewing facilities.	12	5	2	1	2
I improved my working methods thanks to the project.	6	9	7		

Although one project objective is to relate to previous lectures and to mobilize knowledge and skills gained during the bachelor studies, it was not effective and rather seen as a new learning experience, although some students have enjoyed the project as an experience to deepen the different notions of program seen and learned during lectures. We were surprised with the relatively poor use of reviewing.

Students' comment. Students appreciated that each project phase has been explained from experience and through examples. Students have been convinced of the usefulness of the different phases performed in a software project and that it might be applied to other type of projects. Generally speaking, students prefer project to labs. Using on-line tutorials as a learning support is appreciated, but some students complain about the quality of some tutorials written by the teacher. A forum could be useful to share knowledge and help others people. Shared documents could be an alternative to mail exchange and might trigger the use of reviewing facilities that some students misused. Students asked to be exposed to a whole picture of the project at the beginning and to start the project having all project documents at their disposal. Some students found the work load too heavy and time devoted to the project too short. As a student said, all students learned something during the project, and some students have learned more than others!

5. Conclusion

The research question aims to see how students' self-assessment and external assessment [by a teacher] are correlated. This is not true for topics not addressed in the curriculum or unknown by students. For more classical topics, assessments are correlated roughly for the half of the study population. However, the study is a suffering from a bias due to the learning process: deliverables go through a Author-Reader cycle that leads to improve them sufficiently to achieve a Largely Achieved or Fully Achieved level but only "good" students are aware of the help provided by the teacher at each iteration. Hence "good" students under assess themselves whereas "normal" students over assess themselves considering that the resulting deliverable is a witness of their achievement level. The bias invalidates partially the experiment that has to be set again outside of a learning situation.

Questionnaire and students-authors contribution indicates that the system favors knowledge building, encourage students to work in an active way, develop autonomy and success feeling, improve assessment and may develop mutual help; what is expected in a successful project-based learning situation. Process learning seems to be effective for requirements, design and building but we need to improve the system for the ENG.7 SW integration and ENG.8 SW testing process.

6. Acknowledgments

We thank all the students of the 2014-2015 final year of Bachelor in Computer Science to their agreement to participate to this study.

References

[Duf96] T. M. Duffy, D. J. Cunningham. Constructivism : Implications for the design and delivery of instruction. *Handbook of Research for Educational Communications and Technology*, MacMillan, 1996.

[Dwy94] D. Dwyer. Apple Classrooms of Tomorrow: What we have learned. *Educational Leadership*, 54(7), 1994.

[15504] ISO/IEC 15504:2004. *Information technology -- Process assessment*. ISO, Geneva, 2006.

[Tar98] J. Tardif. *Intégrer les nouvelles technologies de l'information – Quel cadre pédagogique ?*. ESF, 1998